UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniaría Informática

TRABAJO FIN DE GRADO

Aplicación móvil orientada a la gestión y distribución de combustible a clientes finales

Miguel Montero Rojas
Tutor: David González
Ponente: David Domínguez

Junio 2016

Aplicación móvil orientada a la gestión y distribución de combustible a clientes finales

AUTOR: Miguel Montero Rojas
TUTOR: David González

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2016

Resumen

El presente **Trabajo Fin de Grado** está centrado en el diseño, desarrollo y pruebas de una aplicación dirigida a una importante compañía petrolífera (de aquí en adelante, ***La Compañía***). La implementación del proyecto se ha llevado a cabo en la empresa de desarrollo de software ***Athelia Solutions***.

El proyecto en cuestión se denomina ***Django TMV Energéticos***. Es una aplicación que permite mejorar la gestión y distribución de combustible líquido a granel para clientes particulares. Estas operaciones de distribución son llevadas a cabo por parte de empresas de transporte subcontratadas por *La Compañía*, por tanto, podemos decir que el usuario final de esta herramienta que hemos creado es el conductor del camión que transporta el combustible.

La aplicación va instalada en un terminal móvil industrial. A través de ella, el conductor recibirá las rutas de reparto diarias o ventas en ruta provenientes de un *middleware* de la compañía petrolífera. Así mismo, el programa es capaz de comunicarse con un caudalímetro (unidad instalada en el camión que mide el totalizador de litros que dispensa la cisterna) gracias al cual podemos ver tanto la cantidad que se está suministrado como la temperatura del combustible. Foto en Anexo C.

Además del terminal móvil, el usuario también dispone de una impresora de calor con la que podrá imprimir facturas y notas de entrega de los pedidos que se realicen.

La aplicación desarrollada, que se describe en este TFG, sustituirá a una versión previa de la misma que requería una renovación total, debido a un software obsoleto y la inclusión de nuevas funcionalidades. De esta forma, proporcionará a *Django TMV Energéticos* una calidad superior y una mejor optimización de los recursos.

Abstract

This Bachelor Thesis is focused on the design, development and testing of an application for a major oil company, hereinafter *La Compañia*. The project implementation took place in the software development company Athelia Solutions.

The name of this project is *Django TMV Energéticos*. It is an application that allows the management and distribution of bulk liquid fuel to private customers. These distribution operations are carried out by transportation companies subcontracted by *La Compañia*. Therefore, we can say that the final user of this tool is the driver of the oil truck.

Through this application, which is installed on industrial handheld devices, the driver will receive delivery routes on a daily basis from a middleware of the oil company. Additionally, the program is able to communicate with a flow meter (a device installed on

the truck that measures the number of liters dispensed by the tank), thanks to which we can know not only the amount of fuel dispensed, but the temperature of it as well.

In addition to the handheld device, the user also has a thermal printer used to print invoices and the delivery notes.

The application developed on this Bachelor Thesis will replace a previous version that was completely outdated and needed a full renovation due to deprecated software and lack of functionality. Through this application, we'll provide *Django TMV Energéticos* a superior quality and a better optimization of the resources.

Palabras clave

Terminal móvil industrial, *La Compañía*, caudalímetro, venta en ruta, nota de entrega, *Django TMV Energéticos, Athelia Solutions*.

Keywords

Industrial Handheld Device, flow meter, route sale, delivery note, *Django TMV Energéticos, Athelia Solutions*.

Agradecimientos

A *Athelia Solutions*, por haberme dado la oportunidad de crear este TFG con ellos.

A mis compañeros de trabajo, por su inestimable ayuda.

A todos aquellos entusiastas que un día decidieron buscar en la informática un modo de vida.

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

Para introducir adecuadamente este TFG, debemos retroceder hasta septiembre de 2015, cuando la Escuela Politécnica me ofreció la posibilidad de realizar las prácticas con la empresa *Athelia Solutions*, como parte de mi formación académica. Después de unas semanas de adaptación, empecé a trabajar en un proyecto que acababa de comenzar su fase de desarrollo: una aplicación móvil basada en C# .Net y Windows Mobile. Este proyecto se había denominado *Django TMV Energéticos*.

Se trataba de un proyecto incipiente y prometedor. Desde el principio podría ver cómo iba a crecer una aplicación real en un entorno laboral, y así ganar experiencia.

Terminado el periodo de prácticas, me ofrecieron una renovación de la beca y continuar con este proyecto. Fue entonces cuando vi una buena oportunidad para realizar el Trabajo de Fin de Grado con *Athelia* y poder compaginar los estudios con la vida laboral.

En primer lugar, vamos a explicar qué significa el nombre del proyecto, *Django TMV Energéticos*. Está claro que la palabra *Energético*s nos sitúa en el entorno del cliente al que va destinada la aplicación, ya que es una importante compañía del sector de la energía, pero la palabra *Django (Jean Baptiste Django Reinhardt)* no es el "desencadenante" de un película de Tarantino, sino el nombre de un framework de trabajo desarrollado por *Athelia* para la creación de aplicaciones móviles y que constituye la base de muchos proyectos que tiene la empresa. Ya hablaremos de Django más en profundidad a lo largo de esta memoria. Por último las siglas TMV significan Terminal Móvil de Ventas, un código identificador que tiene cada conductor.

El concepto general en el que se basa la aplicación es simple: se trata de una herramienta para gestionar el suministro de combustible líquido a clientes particulares. En ella, el usuario, es decir, el conductor del camión cisterna, podrá iniciar un viaje a partir del cual dispondrá de varias rutas de suministro o ventas en ruta, donde están incluidos los pedidos que los clientes de *La Compañía* han solicitado.

La venta en ruta comienza con una carga inicial de los productos necesarios para realizar todos los pedidos de dicha ruta. La cisterna del camión está dividida en compartimentos, donde se pueden cargar distintos tipos de combustible. Y en cada orden de suministro, se encuentra toda la información acerca del cliente (nombre, dirección, DNI…), del tipo de producto que ha solicitado (tipo de combustible, cantidad, precio unitario…) y de los métodos de pago de que dispone el receptor de la mercancía.

1.1 Motivación

La oportunidad de realizar el TFG en un entorno profesional, y conocer así el funcionamiento de la empresa, junto al hecho de poder compaginar los estudios con la vida laboral, supusieron para mí una gran motivación.

Para *La Compañía,* la motivación también era evidente, pues esta aplicación permitiría aumentar la capacidad de gestión de su negocio, con las ventajas que ofrece un software modernizado.

Así, al tiempo que *Athelia* gana un nuevo contrato con este grupo petrolífero, está favoreciendo que el mantenimiento de la aplicación, al tener un programa más actual, sea más sencillo.

1.2 Objetivos

El objetivo es crear una aplicación que irá instalada en terminales móviles industriales, desarrollada en Windows Mobile, para la gestión y distribución de combustible líquido a clientes finales.

Se persigue una aplicación adaptada a la necesidad del cliente, con las mismas funcionalidades que tenia la versión anterior, incorporando otras nuevas, pero con un software moderno y más fiable que mejorará tanto la usabilidad como el mantenimiento de la misma.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del Arte**: En este apartado se da una visión del contexto en el que se sitúa el trabajo y se detallan las herramientas utilizadas para ello.
- **Diseño:** En este apartado se analiza el proceso por el que debe pasar el proyecto antes de empezar a desarrollarse. También se introduce el sistema *Django* en el que se basa la aplicación y se detalla la arquitectura y la estructura interna que tiene dicha aplicación.
- **Desarrollo**: En este apartado se explica el funcionamiento de la aplicación, así como algunos detalles de cómo ha sido programada.
- **Integración, pruebas y resultados**: En este punto contamos los tipos de pruebas y los tests que se han realizado para garantizar la calidad del software.
- **Conclusiones**: En este punto extraeremos las ideas básicas surgidas tras finalizar el trabajo.
- **Anexo**

2 Estado del arte

En este apartado vamos a ofrecer un estudio detallado de los aspectos fundamentales del trabajo realizado para familiarizar al lector con el área de estudio que estamos tratando y con el entorno de trabajo.

2.1 Athelia Solutions

Las nuevas tecnologías han irrumpido en prácticamente todos los aspectos de nuestras vidas, tanto a nivel profesional como personal.

Athelia es una compañía que está especializada en aplicar tecnologías digitales en la industria, con el objetivo de transformar procesos, activos y personas para incrementar el rendimiento de sus negocios. A través de aéreas fundamentales, como análisis y conversión de datos en conocimiento (Big Data), movilidad, internet de las cosas, etc., ayudan a diversas industrias a realizar sus tareas de forma más eficiente, así como a mejorar y transformar sus procesos y modelos de negocio, creando de este modo nuevas interacciones con clientes y aumentando los beneficios. [1]

2.2 Versión Anterior a Energéticos

Antes de crearse el proyecto *Django TMV Energéticos*, existía una herramienta desarrollada por una empresa del sector, ajena a *Athelia,* y utilizada por *La Compañía*.
Dado los problemas que esa aplicación presentaba, *La Compañía* optó por nombrar a *Athelia* su *partner* de movilidad, con quien ya venía colaborando para llevar a cabo otros proyectos.

Un tiempo más tarde, se decidió dar un paso adelante: crear una nueva aplicación con las mismas funcionalidades, pero con un software moderno y propio, de simple manejo, con el que se evitaran antiguos problemas y más fácil de mantener.

Por este motivo, se apostó por *Django TMV Energéticos* como solución de futuro.

2.3 Terminales Móviles Industriales

En este apartado vamos a presentar los modelos de terminal móvil que se han utilizado en el desarrollo del proyecto, y donde va a ser instalada la aplicación para su posterior uso. Estos terminales móviles son de carácter industrial, más robustos que los móviles que a menudo se utilizan en la vida cotidiana. Están pensados para trabajos de campo, resistiendo caídas de hasta dos metros de altura y un uso menos delicado.

Estos terminales se conectan al ordenador mediante una "cuna" que le sirve de carga.

El modelo de terminal que se van a utilizar es Honeywell CN70 de Intermec [2]. Algunas de sus características son:

- Microprocesador Arquitectura: Texas Instruments® OMAP 1 GHz
- Microsoft Windows Embedded Handheld 6.5
- Batería 3.7 V, 4000 mAh
- 512 MB de RAM (aprox. 384 MB libres)
- 1 GB de memoria Flash (aprox. 800 MB libres)
- Pantalla VGA de 3.5" 480 x 640 píxeles
- WPAN: Bluetooth™'II integrado, versión 2.1+EDR

Figura 2-1: CN70

Otro modelo de terminal que también se va a utilizar es el Honeywell CN3 de Intermec. Aunque este dispositivo es algo más antiguo que el anterior, la aplicación también debe estar preparada para él, ya que *La Compañía* aún dispone de algunos ejemplares de este modelo. Sus principales características son:

- Microprocesador Intel® XScale PXA270, 520 MHz
- Windows Mobile 5.0
- Bateria 2200 mAh, 3.7V
- Memoria RAM 128 MB
- Pantalla QVGA 3.5″(89 mm) (240 x 320 pixel) 64K

Figura 2-2: CN3

2.4 Impresora Térmica

Recordamos que nuestra aplicación también es capaz de generar informes en papel de las operaciones que realizamos; así sucede, por ejemplo, al abrir una venta en ruta, al realizar un suministro, etc. (Más adelante explicaremos todo acerca de las funcionalidades de que dispone). Para ello es necesario un hardware adicional que, en este caso, es una impresora térmica, capaz extraer albaranes de venta, notas de entrega de un suministro, liquidación…

El modelo que se va a utilizar es una PB-51 de Intermec: impresora portátil de recibos, robusta, que está preparada para golpes y caídas habituales en los trabajos de campo.

Figura 2-3: Impresora PB-51

2.5 Cuna del Vehículo

Para conectar el dispositivo móvil donde va instalada la aplicación con el caudalímetro del camión, se utiliza una "cuna" que, a la vez de dar soporte al terminal, proporciona la conectividad a través de un puerto serie. Algunas de las características de la cuna del camión son:

- Retiene equipo móvil en el entorno del vehículo.
- Fácil extracción con una sola mano del dispositivo.
- proporciona USB Anfitrión (HDB15M) y RS -232 recipientes (DB9M) .
- Proporciona 5V DC, el RS - 232 para la alimentación de 9 pines periféricos.
- Adaptadores

Figura 2-4: cuna del vehículo

2.6 Herramientas utilizadas

A continuación, vamos a presentar las herramientas que hemos utilizado para el desarrollo de la aplicación. Algunas de ellas, como los sistemas de gestión *Confluence* y *Jira*, forman parte de la organización interna del departamento de proyectos en *Athelia*.

2.6.1 Microsoft Visual Studio 2008

Visual Studio es un entorno de programación para Windows que permite el desarrollo de aplicaciones en múltiples lenguajes; en nuestro caso utilizaremos C# .NET, un lenguaje de programación orientado a objetos, creado por Microsoft. El manejo es visual y cuenta con una gran cantidad de extensiones de ayuda al programador.

Alguno de los *plugins* utilizados a lo largo del proyecto son:

- *StyleCop*: un complemento que analiza el código para que se adapte a unas reglas de estilo, y así tener una organización común en todos los ficheros.
- *ReSharper*: una herramienta de ayuda al desarrollador, que permite realizar automáticamente diversas tareas, como autogenerar trozos de código, optimizar funcionalidades, etc.

2.6.2 Jira y SVN

Jira es una herramienta web para la gestión de proyectos. Con ella se puede gestionar los tiempos empleados, dividiendo en tareas y sub-tareas los distintos apartados y funcionalidades de tu proyecto. Con esta aplicación, también se pueden organizar incidencias ocasionadas en una aplicación cuando ésta se encuentra en estado de producción, o errores que se van encontrando.

Está vinculada con *Visual Studio* y SVN para el control de versiones. De esa manera, los cambios realizados se organizarían dentro de las tareas del proyecto, pudiéndose comentar los mismos. Así, a otros programadores les resultará más fácil entender las modificaciones realizadas.

2.6.3 Confluence

Confluence es un software web de colaboración para equipos de trabajo. Se utiliza para compartir documentación de proyectos, de modo que exista un sitio donde los documentos técnicos, documentos de desarrollo, especificaciones, etc. estén organizados y fácilmente accesibles para todos los integrantes.

2.6.4 Jenkins

Jenkins es un software que proporciona mecanismos de integración continua durante el desarrollo. La integración continua es un modelo informático basado en realizar compilaciones y ejecuciones de pruebas del proyecto de forma automática. Agiliza procesos, como la generación de paquetes de instalación y la ejecución de bancos de pruebas.

2.6.5 MyMobiler

MyMobiler es un programa que permite contralar la pantalla de un terminal con Windows

Mobile, de forma que pueda ser utilizado a través del ordenador, usando el ratón o el teclado del PC. Resulta útil a la hora de copiar archivos entre dispositivos o para escribir de forma más ágil.

2.7 El caudalímetro de la cisterna del camión

El caudalímetro es una unidad instalada en la cisterna del camión que mide el totalizador de litros que dispensa la cisterna. Funciona como el contador de agua que tenemos todos instalado en nuestras casas, es decir, muestra los litros de combustible que pasan a través de la manguera, tanto a la hora de cargar la cisterna como a la hora de suministrar.

La aplicación obtendrá la conexión con el caudalímetro a través de la cuna del camón.

3 Diseño

El diseño es una parte fundamental en un proyecto Software. Es la fase en que se define la forma que va a tener el producto final, las tareas en las que se va a dividir, las metodologías y técnicas que se van a usar y el estilo que va a tener la aplicación.

3.1 Plan de Desarrollo

Antes de empezar a desarrollar un programa, es necesario una planificación, es decir, un estudio del tiempo que va a llevar desarrollarlo. Para ello, debemos antes analizarlo y dividir en tareas cada una de las funcionalidades que va a tener. De esa forma, es más fácil estimar cuánto tiempo va a utilizarse para realizar todo el trabajo.

Para hacer esto posible, utilizamos la herramienta web *Jira* de la que ya hemos hablado anteriormente. En la Figura 3-1 se muestra una captura de una parte del plan de desarrollo del proyecto en *Jira*. En ella se muestran algunas tareas principales, llamadas *Epic*, y un conjunto de sub-tareas que dependen de ella. También se puede apreciar el estado en el que se encuentran cada una de las tareas, según la columna en que se halle (de izquierda a derecha). Estos estados de desarrollo pueden ser: "pendiente", "en proceso", "para revisar" y "terminado". En nuestro caso, todas las sub-tareas se encuentran revisadas y terminadas.

Pero la herramienta Jira tiene mucho más uso que el descrito: si se pincha en una de las tareas, se puede ver el tiempo que se estimó para realizarla, qué persona es la encargada de desarrollar la tarea y cuánto tiempo está empleando para ello. Así, se puede controlar en qué tarea están empleando el tiempo los trabajadores, ya que después de cada día de trabajo es responsabilidad del desarrollador introducir el tiempo en cada una de las tareas en las que ha estado trabajando.

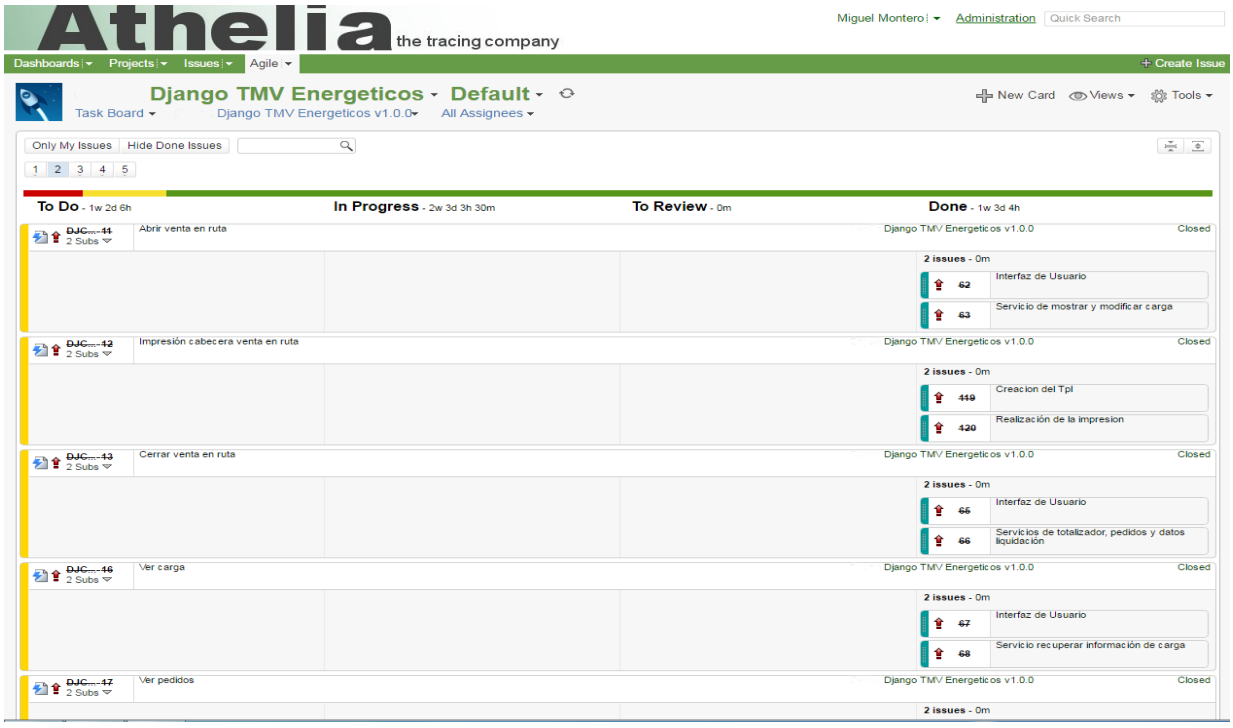


Figura 3-1: TaskBoard de Jira

3.2 Django

Ahora toca hablar un poco más en profundidad: en qué consiste *Django*. Durante las primeras semanas de trabajo, buena parte del tiempo lo dediqué a comprender el funcionamiento de este software, fundamental para poder desarrollar la aplicación final.

Las aplicaciones de gestión de combustible y trazabilidad de activos guardan muchas similitudes entre sí, aunque pertenezcan a clientes distintos. Por ejemplo, las instalaciones o las órdenes de trabajo de las compañías con las que se trabaja son intrínsecamente iguales o muy parecidas. *Django* aprovecha estas similitudes inteligentemente para crear un entorno de desarrollo que integre en sus cimientos esas características.

Más concretamente, el objetivo de *Django* es crear un entorno común para el desarrollo de aplicaciones en dispositivos móviles industriales, aprovechando las similitudes que presentan. La ventaja que ofrece contar con un "esqueleto" prediseñado, es poder partir de una base al elaborar una nueva aplicación, lo que disminuye en gran medida el tiempo de desarrollo y aumenta la eficiencia. De esta forma, al disminuir el tiempo de desarrollo, también es menor el coste del proyecto. Un entorno común también facilita la detección y corrección de errores durante el mantenimiento del producto.

Somos conscientes del tipo de usuarios a los que van destinadas la aplicaciones que se desarrollan con *Django*. Se utilizan en entornos industriales, de campo, donde están presentes otros tipos de herramientas y trabajos más manuales o el uso de guantes; por eso es necesario adaptarse a este tipo de usabilidad, en la cual la persona que maneja la aplicación no es experta en informática. De esta forma, los botones y todos los elementos visuales de la aplicación, como listas, títulos, iconos, etc. son grandes, claros y directos.

En resumen, *Django* no es un producto final. No se trata de un paquete software listo para ser usado por el cliente. Lo que *Django* busca es reducir el esfuerzo de desarrollo mediante la unión de componentes existentes en lugar de reescribir unos nuevos que ya han sido programados previamente.

3.2.1 Django Mobile Client

El sistema Django es, por un lado, una capa por encima de Windows Mobile para poder programar aplicaciones (ya sea en elementos estructurales como visuales: interfaz de usuario) y, por otro lado, una parte de *Backend* ("el lado del servidor"), que consta de varios servicios. Más adelante, seguiremos hablando de esto último en el apartado de Arquitectura de la Aplicación.

En este punto nos centraremos en explicar cómo se estructura Django a la hora de crear un programa "en el lado del terminal". *Django Mobile Client* es un componente que nos facilita el acceso a herramientas que se usan habitualmente. Vamos a describir alguna de sus partes:

1. **Modelo de Aplicación**: una serie de entidades básicas con las que trabaja todo sistema de *Django*. Vamos a enumerar las más utilizadas en nuestra aplicación:

    - *Route* (ruta)
    - *WorckOrder* (orden de trabajo)
    - *Site* (lugar o emplazamiento físico)
    - *User* (usuario)
    - *UserGroup* (grupo de usuarios)
    - *Referential* (referencial, se utiliza para guardar datos asociados a una tarea)

Estos objetos existen en todas la aplicaciones de *Django* y se adaptan a cada caso, aunque siempre de una manera muy similar. También hacen referencia a las tablas de la base de datos de *Django* de la que también hablaremos más adelante.

2. **Workflows**: son los flujos de ejecución que lleva la aplicación. Su función es el mapeo de conexiones entre tareas mediante nodos. Veamos un ejemplo:

```
<!-- Driver Signature -->
<node id="driver-signature-node" element-id="driver-signature-module">
  <exit return-value="ok"  destination="select-main-menu-node" />
  <exit return-value="back"  destination="driver-selection-node" />
  <exit return-value="error"  destination="driver-selection-node" />
</node>
```

Como se puede ver, los nodos se conectan entre sí con valores de retorno, que indican a qué nuevo nodo les conduce la aplicación. El valor de `element-id` indica qué tarea o módulo debe ejecutarse.

Vamos a poner un ejemplo visual para que quede aún más claro cómo se relacionan los nodos con los módulos:

```
<!-- Driver Signature-->
<module id="driver-signature-module" type="Django.TMV.Energeticos.App.Elements.Modules.Login.DriverSignatureModule,
        Django.TMV.Energeticos.App">
  <properties>
    <property key="name" value="Driver's Signature"/>
    <property key="cancelButtonText" value="Cancel"/>
    <property key="okButtonText" value="Accept"/>
  </properties>
</module>
```

De esta forma, existe un fichero principal de nodos que indican el flujo de la aplicación (*workflow-modules.xml*) y otro (*modules.xml*), con los módulos de esos nodos, que relacionan cada uno de ellos con su correspondiente fichero de código, que es quien realmente dicta qué debe hacer la tarea. También desde el *module* se pueden inicializar algunas propiedades, como el nombre de la tarea o el texto que llevan los botones.

A parte de los *modules*, existen ficheros de *workflow* alternativos, que parten del principal, llamados *tasks*. Se utilizan cuando va a dar comienzo una gestión que terminará con un cambio importante, el cual deberá notificarse al servidor. Esta notificación se realiza a través de una *finished-order*. La *finished-order* es un mensaje xml con toda la información recopilada de la *task* en cuestión, que se enviará a una cola de mensajes donde el servidor la recogerá y sincronizará los datos adecuadamente. Cada fichero de flujo de nodos de una *task* tiene que tener asociado uno de tareas correspondiente, igual que los módulos.

3.3 Modelo de la Aplicación

Ahora que conocemos mejor el funcionamiento de *Django,* podemos explicar cómo hemos organizado el desarrollo de la aplicación.

3.3.1 Arquitectura de la aplicación

Vamos a comentar por encima la arquitectura general que tiene la aplicación. No nos detendremos demasiado en este tema, ya que la parte de *Backend* de la aplicación no es en la que se centra este TFG, aunque sí es importante tener una idea de en qué consiste.

La arquitectura de la aplicación se sostiene sobre tres partes fundamentales, según muestra la siguiente imagen:

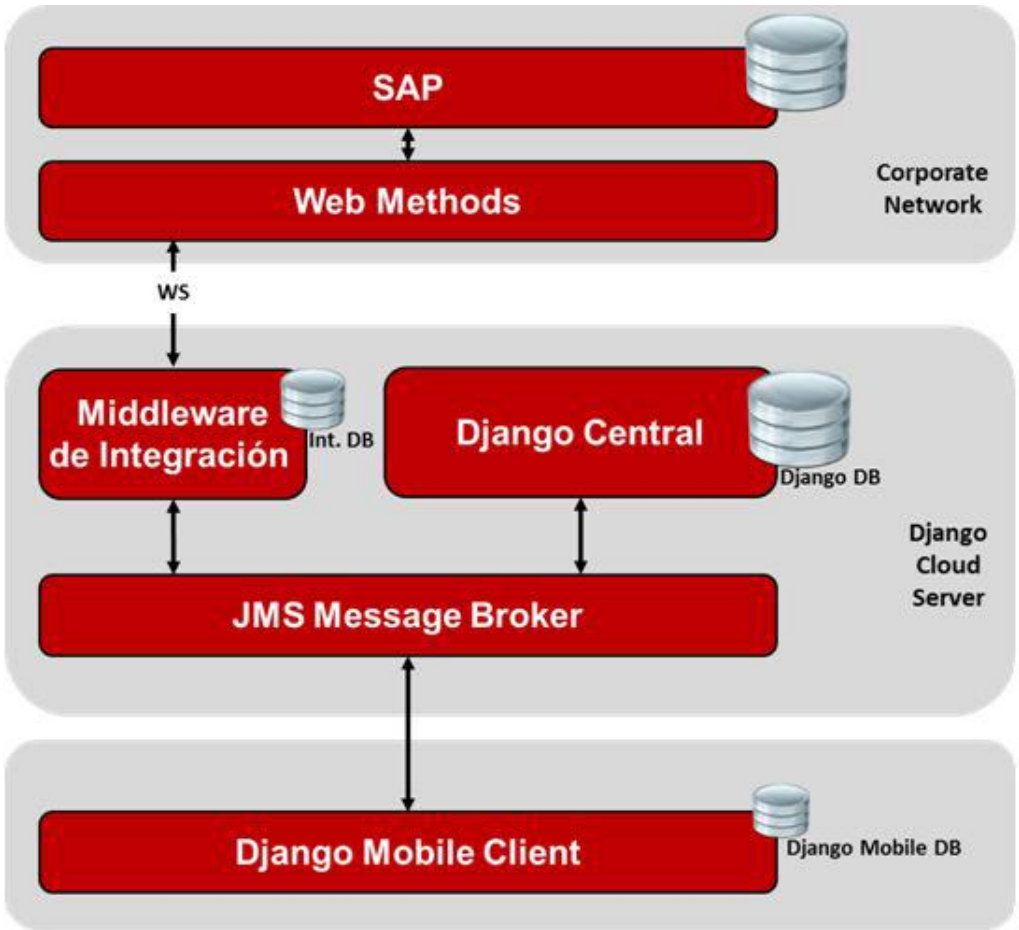


Figura 3-2: Arquitectura de la aplicación

La primera es la parte móvil, es decir, las órdenes que manda el usuario de la aplicación desde el dispositivo cada vez que realiza una tarea (*Django* Mobile Cliente). La aplicación dispone de una base de datos local, gestionada con SQLite, donde se encuentra toda la información necesaria.

La segunda parte es el servidor de *Django.* A través de un bróker de mensajería (colas de mensajes) se comunican el servidor y el terminal. En esta parte existen dos módulos:

- ***Django* Central**: encargado de sincronizar la información entre los terminales y el servidor en ambas direcciones. También es el encargado de almacenar todos los datos de la aplicación y asegurar la persistencia. La base de datos de *Django* sigue un modelo definido que no varía de una aplicación a otra.
- **Integración**: este módulo es el encargado del intercambio de información entre el sistema *Django* y un sistema externo, con el que se comunica a través un *Web Service*, con protocolo SOAP.

La tercera parte es ese sistema externo al que acabamos de aludir, que controla *La Compañía*. Desde aquí se recibe toda la información acerca de los clientes, ventas en ruta, suministros, etc.

3.3.2 Entidades Básicas

Ahora volvemos a la parte de *Django Mobile Client* ("el lado del terminal") para explicar cómo ha sido el modelado de los objetos básicos que se han utilizado para nuestra aplicación en concreto. Ya hemos hablado que *Django* tiene unos objetos comunes que se utilizan en todas sus aplicaciones, pero después, dentro de cada una en particular, se modelan otras entidades nuevas, más especificas, que utilizan la información de los objetos de *Django*. Vamos a enumerar algunas de nuestras clases de datos más importantes:

- **RouteSale**: venta en ruta. Es la clase donde se aporta toda la información de la cabecera de la ruta (código de ruta, fecha de inicio, fecha fin, estado…).
- **Supply**: suministro. Es el tipo de datos donde se encuentra la información de un pedido, es decir, el cliente que lo ha solicitado, el producto, la cantidad, métodos de pago, etc.
- **TruckLoad**: carga del camión. En esta entidad se almacena la información sobre la carga inicial de un producto en la cisterna del camión (tipo de producto, cantidad de carga y temperatura).
- **Product**: producto. Indica el tipo de combustible (si es gasóleo A, gasóleo B, Agromax…) y alguna información de negocio importante.
- **Driver**: conductor. Guarda la información del conductor del camión: nombre, identificador de negocio…
- **Customer**: cliente. Almacena información sobre el cliente que ha solicitado un suministro: nombre, dirección, NIF, etc.

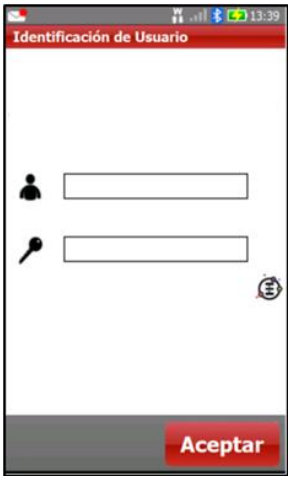
Por cada una de estas clases, existen unos servicios asociados a ellas, es decir, un conjunto de funciones que se encargan de interactuar con la base de datos para pedir o guardar información.

4 Desarrollo

Llegados a este punto, nos queda por explicar lo más importante: la funcionalidad completa de la aplicación.

4.1 Acceso a la aplicación

Nada más ejecutar la aplicación nos aparecerá la pantalla de identificación:



En el espacio en blanco de la parte superior aparecerá el logo de la empresa a la que va destinada esta herramienta. Lo primero que nos pide es una identificación. Para poder entrar, se debe introducir la información de un usuario existente en el sistema. En este caso es necesario la matrícula del camión con el que se va a trabajar.

Una vez introducidos nombre de usuario y contraseña, se accede a la aplicación pulsando "Aceptar".

Figura 4-1: login

Pulsando en la barra de título, se solicitará la contraseña de administración para acceder al menú de servicios o para salir de la aplicación:

Figura 4-2: Menú de administración

Nota: La aplicación está pensada para que ningún usuario pueda salir de ella, excepto un administrador, ya que va instalada en un dispositivo móvil que a todos los efectos funciona como un "ordenador de bolsillo" y solo pueda ser utilizado en un entorno de trabajo.

Nada más hacer *login*, la aplicación comprueba si existe base de datos local. De no ser así, comienza a descargarla desde Django Central. Solo bajarán los datos necesarios para el camión que hemos identificado.

4.2 Menú Principal

Una vez estemos dentro de la aplicación, lo primero que nos solicitará es el nombre del conductor y su firma, para poder acceder al menú principal e iniciar un viaje.

Después, accederemos al menú principal donde tendremos acceso a una serie de servicios (muy parecidos a los del menú de administración) y al inicio de viaje o inicio de actividad.

Figura 4-3: Menú Principal e Inicio de Actividad

4.2.1 Inicio de actividad

Al empezar el inicio de viaje, la aplicación intentará conectarse con el caudalímetro del camión para darnos el total de litros que se han dispensado hasta el momento (historial de litros).

En caso de no existir conexión con el caudalímetro, el usuario deberá introducir los datos (historial de litros) manualmente. Así mismo será necesario introducir el número de kilómetro del vehículo indicado en el cuentakilómetros.

Por último, después de haber iniciado el viaje, llegamos al menú principal desde donde tendremos acceso al listado de ventas en ruta disponibles.

4.2.2 Lista de Ventas en Ruta

Al entrar en la opción de ***Lista de Ventas en Ruta*** aparecerán todas y cada una de las Ventas en Ruta disponibles para el código TMV del conductor.

Por defecto, siempre se muestran las rutas en estado "Abierta" y en estado "Pendiente", y quedan sin mostrarse las ya "Cerradas", "Concluidas" y las "Rechazadas", aunque se podrán activar pulsando su botón correspondiente. La barra de tareas, en la parte inferior de la pantalla, contiene una serie de botones correspondientes a los estados que puede tener la ruta y que actúan de filtro de la lista.

En cada una de las líneas se mostrará el código de la ruta, la fecha y el almacén (Centro de Carga).

En la parte izquierda de cada línea aparecerá el estado de la ruta:

- **Abierta** Ruta para la que ya ha sido cargado físicamente el combustible necesario en el camión, y que ya se encuentra realizando suministros a clientes. Sólo podrá existir una ruta en este estado.
- **Cerrada** Ruta sobre la cual ya se han efectuado todas las órdenes de suministro a clientes. Aún no se ha efectuado la Conclusión de la Ruta, lo que no permitirá la apertura de nueva ruta.
- **Concluida** Ruta sobre la cual ya se han efectuado todas las órdenes de suministro a clientes y adicionalmente se han registrado los valores de Conclusión de Ruta.
- **Pendiente** Ruta pendiente de abrir, con datos de la carga física ya efectuada.
- **Rechazada** Ruta rechazada completamente.

Figura 4-4: Lista de ventas en ruta

Es necesario seleccionar una de las ventas en ruta y pulsar el botón de ***Aceptar*** para acceder al menú de venta en ruta que correspanda.

4.3 Menú Ventas en Ruta

En función del estado de la ruta que se haya seleccionado, se abre un menú de venta en ruta u otro con sus respectivas acciones.

4.3.1 Menú de venta en ruta Pendiente

Este es el menú que se muestra en el caso de seleccionar una venta en ruta cuyo estado se encuentra Inactiva (Pendiente). Las operaciones posibles que se pueden realizar son:

- **Abrir la venta en ruta**. No se puede abrir nueva venta en ruta si existe una aún no Concluida.
- **Ver la carga de la cisterna**
- **Ver los pedidos de la ruta.** Muestra la lista de pedidos, pero no se pueden realizar sin abrir la ruta.
- **Rechazar venta en ruta y sus pedidos**
- **Imprimir los pedidos de la ruta**

Figura 4-5: Menú venta en ruta Pendiente

4.3.1.1 Abrir Venta en Ruta

Mediante esta opción, se procede a la apertura de la venta en ruta para poder realizar los suministros de los pedidos. Antes de abrir la Ruta Pendiente, todas las demás han de estar Cerradas y Concluidas.

En el proceso de apertura de la Venta en Ruta se muestra una primera pantalla con la información relativa a los datos de la carga de productos efectuada anteriormente. En caso de no poder obtener datos reales de la carga efectuada, se permitirá su introducción manual. Aparecerá una pantalla con el valor de la cantidad prevista y temperatura a la que se encuentra. Al pulsar sobre la cantidad prevista o temperatura aparece, el teclado numérico para modificar dichos valores. Pulsar el botón de ***Aceptar*** para validar la modificación.

Se validará que la cantidad introducida no difiera en ± un porcentaje admitido de configuración sobre el dato inicial. En cado de que la cantidad sea cero, se podrá introducir cualquier valor.

A continuación se solicitará la composición del Compartimento de la Cisterna. Mediante el botón + se puede agregar un nuevo compartimento. Mediante el botón – se puede eliminar un compartimento agregado erróneamente.

Al agregar un nuevo compartimento mediante el botón +, aparecerá una pantalla para elegir: el compartimento (de una lista previamente configurada), el producto (de los posibles productos existentes en la carga), y la cantidad del producto.

Se controlará que la suma de las cantidades de los productos de cada compartimento coincida con el total de la carga.

Por último, y antes de pasar a imprimir la cabecera del albarán de venta en ruta, se muestra una pantalla con el resumen de todos los datos de la apertura de ruta. Después de pulsar ***Aceptar***, se guardan todos los datos de esta tarea y se envían al servidor mediante una *finished-order*.

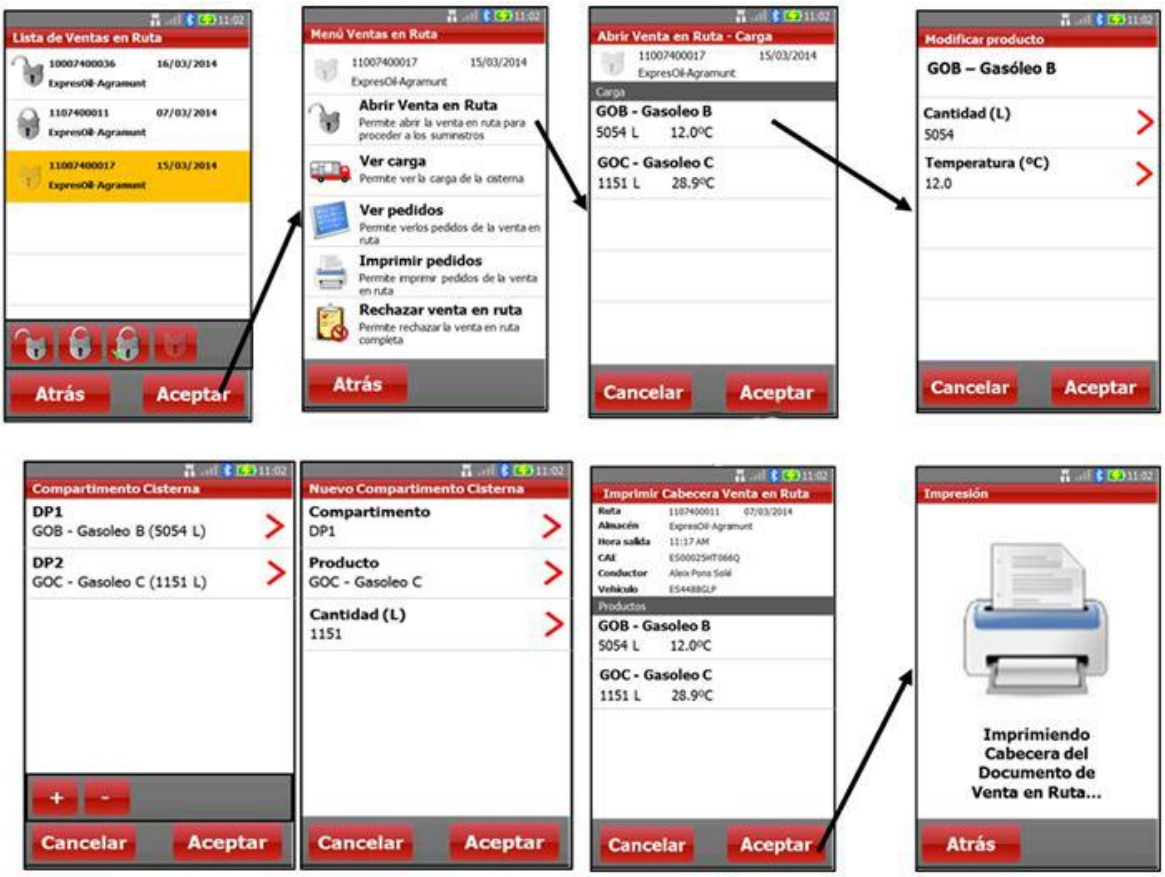


Figura 4-6: Apertura de Venta en Ruta

Un ejemplo de albarán de venta en ruta se encuentra en el Anexo A.

4.3.1.2 Ver Carga

Esta opción permite visualizar el detalle de la carga de la Venta en Ruta seleccionada. La pantalla mostrará el número de Albarán de Venta en Ruta, su estado y su Almacén, las fechas de inicio y fin, así como el nombre del conductor y la matrícula camión al que ha sido asignada la ruta. Mostrará una lista de los distintos productos cargados en el camión y desglosados entre:

- **Carga**: La cantidad de Carga
- **Pedido**: La cantidad pedida comprometida y, entre paréntesis, el número de pedidos del producto.
- **Existencia**: La cantidad cargada menos lo entregado.
- **Entregado**: La cantidad entregada y, entre paréntesis, el número de pedidos entregados.

Figura 4-7: Ver Carga

4.3.1.3 Ver Pedidos

Mediante esta opción se visualizarán los pedidos que tenga una Venta en Ruta. En la parte superior aparece el estado de la venta en ruta, el código de la ruta y la fecha. La lista se muestra ordenada por Horario Programado. En el caso de que la ruta sea ***pendiente***, únicamente se muestran los pedidos para su visualización.

A continuación aparece la lista de los pedidos de la venta en ruta, donde se muestra la información siguiente:

- Estado:

  Pedido suministrado

  Pedido rechazado

  (En blanco para resto de estados)

- Horario programado
- Nombre del cliente
- Teléfono de contacto
- Población
- Código de pedido
- Producto a suministrar y cantidad en litros

Figura 4-8: Ver Pedidos

En la Parte inferior se mostrarán dos botones para poder realizar filtros sobre la lista de pedidos:

- "S" para ocultar pedidos ya Suministrados
- "R" para ocultar pedidos que han sido Rechazados

Ambos botones, por defecto, aparecerán activados para mostrar todos los pedidos de la Venta en Ruta.

Seleccionando un pedido y pulsando el botón Aceptar se accederá a la acción que corresponda. En este caso: el detalle de los datos del pedido.

4.3.1.4 Imprimir pedidos

Mediante esta opción se puede realizar la impresión de los pedidos pendientes en la Venta en Ruta seleccionada.

4.3.1.5 Rechazar venta en ruta

Al seleccionar la opción Rechazar Venta en Ruta se muestra un mensaje de confirmación al usuario. Una vez confirmado el rechazo de la venta en ruta, dicha ruta aparecerá marcada con la imagen  en el navegador de ventas en ruta.

4.3.2 Menú de Venta en Ruta Abierta

Una vez se ha completado la apertura de Venta en Ruta y ya cambiado el estado de la ruta, se muestra el menú de Venta en Ruta abierta.

Las acciones que se pueden realizar son las mismas que en el menú de Venta en Ruta pendiente, exceptuando el cierre de venta de la ruta, y que ahora sí es posible realizar un suministro.

Para cerrar una ruta es necesario que no exista ningún pedido pendiente de la lista de suministros.

Figura 4-9: Menú Venta en Ruta Abierta

4.3.2.1 Realizar Suministro

Desde la opción de ***Ver Pedidos***, seleccionando uno de ellos y pulsando el botón de ***Aceptar***, se accede al menú del pedido para poder suministrarlo.

Después, pulsar ***Realizar Suministro***.

Figura 4-10: Lista Suministros y Menú del pedido

Para realizar un suministro, primero se verificará si el cliente al que se va a suministrar el combustible, debe firmar la Carta de Consumidor Final.

En caso afirmativo se mostrará un aviso al usuario de que debe realizar el proceso de firma.

Se solicitarán los datos del receptor de la mercancía (Nombre y NIF).

Al final de la carta aparecerá la zona correspondiente a la firma por parte del cliente.

Figura 4-11: Firma carta consumidor final

Desde este momento, se inicia el proceso de suministro. Si existe conexión con el caudalímetro, la cantidad y temperatura se actualizará automáticamente. En caso contrario se especificarán en la pantalla siguiente.

Una vez finalizado el suministro se deberá pulsar el botón ***Aceptar***.

En el caso de existir conexión con el caudalímetro, al terminar el suministro se pasará directamente a la siguiente pantalla.

En la parte de arriba de la pantalla se muestra una cabecera con detalles del cliente que solicitó el pedido.

Figura 4-12: Suministrando pedido

A continuación se muestra la información del suministro referente a los importes y los métodos de pago.

Figura 4-13: Información de cobro y métodos de pago

En función del método de pago seleccionado, la referencia de cobro puede variar:

- En el caso de seleccionar Pago Metálico, la referencia de cobro es un código alfanumérico único que se calcula automáticamente.
- En el caso de seleccionar Pago Tarjeta, la referencia de cobro serán los cuatro últimos dígitos del número de la tarjeta del cliente.
- En cualquier otro caso no hay referencia de cobro.

Por último, el cliente tendrá que firmar la recepción de la mercancía, y se procederá a imprimir la nota de entrega del suministro y la cabecera del albarán de venta en ruta, junto con los suministros que se hayan realizado hasta el momento.

Un ejemplo de albarán de venta en ruta completo se encuentra en el Anexo A, y un ejemplo de nota de entrega en el Anexo B.

Acto seguido, se guardarán los datos de esta tarea y se enviarán al servidor mediante una *finished-order* con toda la información acerca del suministro.

4.3.2.2 Cerrar Venta en Ruta

Para cerrar una Venta en Ruta es necesario que todos los pedidos de la lista estén suministrados o rechazados. Es posible rechazar un suministro de la lista desde el menú de suministro (Figura 4-10), indicando el motivo del rechazo.

Al entrar en la opción de ***Cerrar Venta en Ruta,*** si existe caudalímetro, se realizará la conexión con éste para obtener el totalizador de litros. El valor obtenido no se podrá modificar. En caso de no existir caudalímetro, o en caso de no poder conectar con él, se mostrará la cantidad global totalizada (cantidad inicial totalizada más la suma de suministros realizados).

A continuación, aparecerá una pantalla con información de la ruta acerca de cuántos pedidos existían, y cuántos han sido realizados, tanto en cómputo global como por tipo de producto.

Al pulsar el botón ***Aceptar***, toda la información del cierre de ruta se guardará y se enviará al servidor mediante una *finished-order*.

Posteriormente se procederá a imprimir el albarán de venta en ruta completo: cabecera, cada uno de los suministros realizados y pie, con información de qué cantidad de cada producto se ha vendido y qué cantidad se debe devolver. Este documento estará firmado por el conductor.

Después de imprimir el albarán, se mostrará la liquidación. En ella aparece información sobre el dinero recaudado, tanto ingresos en efectivo como con tarjeta; también la información sobre los litros consumidos desde el inicio hasta el final de la ruta, y los kilómetros recorridos durante la Venta en Ruta.

Después de mostrar la liquidación por pantalla, se podrá imprimir ésta.

Figura 4-14: Cierre de Venta en Ruta

Una vez terminado este proceso, la ruta se encuentra cerrada. Pero eso no es suficiente para abrir una nueva. La ruta debe pasar por un último proceso de cierre efectivo.

4.3.3 Menú de Venta en Ruta Cerrada

Cuando la ruta está cerrada, las únicas acciones que se pueden realizar son:

- Volver a imprimir los documentos que previamente se habían estado generando: suministros realizados, albarán de venta en ruta completo y liquidación.
- Podemos ver también el resumen de la carga. Es la misma pantalla que la que se muestra en la Figura 4-8.
- Ver la lista de pedidos, los cuales estarán suministrados o rechazados. Para los ya suministrados, tenemos la opción de volver a imprimir la nota de entrega de ese suministro.
- Y por último, podemos realizar el cierre efectivo de la ruta.

Figura 4-15: Menú Venta en Ruta Cerrada

4.3.3.1 Cierre Efectivo

Una vez la Venta en Ruta se ha cerrado, se puede proceder al Cierre Efectivo de ésta.

La información que se solicita para el cierre efectivo es la siguiente:

Figura 4-16: Cierre Efectivo

- **Devolución teórica**: será el resultado de sustraer de la Cantidad Inicialmente Cargada la suma de las Cantidades Reales Entregadas a Clientes.
- **Devolución Física**: Una vez realizada la devolución de todo el producto restante contenido en el camión, el conductor deberá introducir la cantidad real descargada.
- **Lugar de Vaciado**: El conductor deberá registrar uno de los siguientes valores: "Centro de Carga" u "Otro".

- **Totalizador (litros)**: mostrará la Cantidad total de litros.
- **Totalizador (Km)**: mostrará la Cantidad indicada en el contador de kilómetros.

Una vez introducidos todos los datos y pulsemos ***Aceptar***, se guardará toda la información y se notificará al servidor mediante una *finished-order*.

El menú de Venta en Ruta Concluida es el mismo que el de Venta en Ruta Cerrada, excepto la opción del cierre efectivo, evidentemente.

4.4 Fin de Actividad

Por último, cuando el conductor haya terminado con todas las ventas en ruta de ese día, se debe proceder al ***cierre de actividad*** o fin del viaje.

Volvemos al menú principal y seleccionamos la opción ***fin de actividad.***

Figura 4-17: Fin de Actividad

El flujo de fin de actividad es el mismo que el de inicio, pero ahora para el cierre de viaje. Se intentará conectar con el caudalímetro para recoger el totalizador de litros. De no existir caudalímetro o no haber podido efectuar la conexión, se pedirán los datos manualmente. Las cantidades no pueden ser menores que en el inicio de viaje.

5 Integración, pruebas y resultados

En este punto vamos a describir los tipos de pruebas que se realizaron para garantizar la calidad del software desarrollado.

Las pruebas no se realizaron todas al final, sino que, conforme se añadía una nueva funcionalidad a la aplicación, ésta se probaba modularmente para testear si su funcionamiento era correcto. Aun así, es necesario llevar a cabo un plan de pruebas final para probar todo en conjunto.

Recordamos que al principio del desarrollo, dividimos la aplicación en tareas según las funcionalidades que tenia la aplicación, y montamos un plan de desarrollo en la herramienta *Jira*. Estas tareas podían encontrarse en diversos estados, como *Pendiente*, *En Desarrollo* o *Para Revisar*. Cuando una tarea se termina y se mueve al estado *Para Revisar*, se designa una nueva persona (distinta al programador que desarrolló la tarea) para testear si ha sido realizada con éxito.

5.1 Plan de Pruebas

Una vez que todas las tareas se han terminado y revisado, se crea un plan de pruebas final, en el que se detallan todos y cada uno de los escenarios posibles por los que puede pasar la aplicación, y se describe cuál es el funcionamiento correcto que debería seguir la aplicación para cada uno de esos escenarios. A continuación se muestra una captura de una parte del plan de pruebas, ya que no ha sido posible adjuntarlo por completo debido a su larga extensión:

TS29	Realizar Suministro	TC29.13	suministrando / conexión con el caudalimetro / boton aceptar
TS29	Realizar Suministro	TC29.14	suministrando / conexión con el caudalimetro / boton cancelar
TS29	Realizar Suministro	TC29.15	suministrando / sin conexión con el caudalimetro/ reintentar si
TS29	Realizar Suministro	TC29.16	suministrando / sin conexión con el caudalimetro/ reintentar no
TS29	Realizar Suministro	TC29.17	suministrado sin conexión con caudalímetro / boton aceptar
TS29	Realizar Suministro	TC29.18	suministrado sin conexión con caudalímetro / boton cancelar
TS29	Realizar Suministro	TC29.19	Informacion de pago
TS29	Realizar Suministro	TC29.20	Informacion de pago / boton aceptar / valorado
TS29	Realizar Suministro	TC29.21	Informacion de pago / boton aceptar / no valorado
TS29	Realizar Suministro	TC29.22	Informacion de pago / boton cancelar

Figura 5-1: Escenarios de prueba

La Figura 5-1 muestra los escenarios de prueba que pueden darse durante el suministro de un pedido.

TC29.13	suministrando / conexión con el caudalimetro / boton aceptar	TC29.13	Conecta con el caudalimetro y rellena los valores. Si pulsa aceptar nos lleva a la informacion de pago.
TC29.14	suministrando / conexión con el caudalimetro / boton cancelar	TC29.14	Cancela el suministro y vuelve a la lista.
TC29.15	suministrando / sin conexión con el caudalimetro/ reintentar si	TC29.15	No ha podido conectar con el caudalímetro e intenta una reconexión.
TC29.16	suministrando / sin conexión con el caudalimetro/ reintentar no	TC29.16	No ha podido conectar con el caudalímetro y pasa a la siguiente pantalla donde deverá introducir los valores de forma manual.
TC29.17	suministrado sin conexión con caudalímetro / boton aceptar	TC29.17	No ha existido conexión con el caudalimetro y se deven introducir los valores de carga manualmente. Se comprobará que los valores introducidos no superan la cantidad carga inicialmente. De no ser así muestra un mensaje de error. Si todo es correcto la aplicación va a la información de pago.
TC29.18	suministrado sin conexión con caudalímetro / boton cancelar	TC29.18	Se cancela el suministro y vuelve a la lista de suministros
TC29.19	Informacion de pago	TC29.19	Muestra informacion sobre el precio total unitario, el total del importe y si el cobro es requerido.
TC29.20	Informacion de pago / boton aceptar / valorado	TC29.20	La aplicación va a la pantalla del método de pago.
TC29.21	Informacion de pago / boton aceptar / no valorado	TC29.21	Si el suministro es no valorado, la aplicación nos lleva directamente a la firma del cliente.
TC29.22	Informacion de pago / boton cancelar	TC29.22	La aplicación nos devulve a la pantalla donde se realizó el suministro con o sin caudalímetro según corresponda.

Figura 5-2: Casos de Prueba

La Figura 5-2 muestra los casos de prueba pertenecientes a los escenarios que mostraba la Figura 5-1.

5.2 Test Unitarios

Los test unitarios o *Unit Test* son pruebas enfocadas al testeo de funciones o métodos de una clase. Para llevar a cabo este proceso de pruebas, se utiliza un *framework* para lenguajes basados en .NET (NUnit), que permite la ejecución de pruebas unitarias directamente en el entorno de desarrollo. Se basa en el concepto de *Assertions* o comprobaciones de valor (*AreEqual*, *AreNotEqual*, *IsTrue*, *IsFalse*, etc.). [3]

Los test unitarios los vamos a aplicar sobre la capa de servicios de que dispone cada uno de los tipos de datos de nuestra aplicación, que definimos en el punto 3.3.2 de este documento.
Estas pruebas unitarias se ejecutarán automáticamente cada vez que se suba un cambio al repositorio de versiones SVN, así, si se hace alguna modificación, sabremos que no hemos estropeado ningún método que existía antes, y que todos los accesos a base de datos siguen funcionando bien.

Los *Unit Test* los hemos llevado a cabo de la siguiente forma: Primero, cogemos la base de datos local de Django y creamos una copia vaciando los datos. Después, creamos un script SQL para insertar los datos necesarios para cada una de las pruebas por separado. Sobre esa base de datos preparada, pasamos los tests y comprobamos que la solución es correcta mediante los *Assertions*.

5.3 Despliegue de la Aplicación en el Terminal

El despliegue e instalación en el terminal es una operación que se sigue durante el desarrollo del software. Una vez compilada la aplicación, se realiza el proceso de despliegue. Para controlar más fácilmente el terminal, se utiliza la herramienta MyMobiler (veáse 2.5.5). *Visual Studio 2008* permite desplegar la aplicación en un terminal conectado

al PC por medio del *Windows Mobile Device Center*. Este programa, además de instalar los drivers correspondientes para comunicarse con el dispositivo, establece una conexión segura con el mismo.

Las labores de prueba se agilizan significativamente gracias al servicio de *log* de la aplicación, que registra cualquier posible error, volcando la información en un fichero de texto. Así, en el momento en que se produce un comportamiento inesperado, se puede acudir a dicho fichero para ver qué ha ocurrido. Existen varios niveles de *log*. Vamos a enumerarlos por nivel de gravedad, de menor a mayor:

- DEBUG
- INFO
- WARNING
- ERROR

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El negocio tecnológico en el ámbito industrial, así como el mercado de los terminales y dispositivos móviles, está creciendo cada vez más, lo que ocasiona un rápido avance.

La adaptación a este crecimiento es de suma importancia para una buena renovación y para un asesoramiento actualizado a los clientes.

Así fue como *Athelia Solutions* propuso a una de las empresas del sector de la energía más importantes de España (*La Compañía*) la renovación completa de una de sus aplicaciones para la gestión y distribución de combustible a clientes finales. Esta actualización supondría un mejor rendimiento de la herramienta, y una mejor usabilidad adaptada a las necesidades del cliente y del usuario gracias al sistema *Django*.

Así surgió la necesidad de crear la aplicación que hemos expuesto en este documento: diseño, desarrollo y pruebas de una nueva aplicación. Diseño y desarrollo llevados a cabo gracias al sistema de trabajo de *Django* y a las herramientas de gestión y organización, como *Jira*, *Jenkins* y *SVN*. Las pruebas aseguraron la calidad del software y un buen funcionamiento conjunto.

6.2 Trabajo futuro

Tras la sustitución de la versión antigua, el producto final ha sido una aplicación plenamente funcional, adaptada a los usuarios y desplegada en un entorno de trabajo real.

Sin embargo, siempre pueden surgir ampliaciones de la funcionalidad de la aplicación, como, por ejemplo, un método de fidelización de los clientes con un sistema tarjeta de puntos similar al que se utiliza en las gasolineras o los supermercados.

Sin duda alguna, éste sería un reto importante y un trabajo de futuro inmediato.

Referencias

[1] *Athelia Solutions* “Soluciones para transformar tu negocio” http://www.athelia.com/

[2] Honeywell Intermec “Sensing and Productivity Solutions ”
http://www.intermec.com.mx/products/cmptrcn70a/index.aspx

[3] Kent Benk “Nunit” http://www.nunit.org/

Glosario

API	Application Programming Interface
La Compañía	Importante empresa Española del sector de la energía a quien va destinada esta aplicación.
TMV	Terminal Móvil de Ventas
Athelia Solutions	Empresa en la que se ha desarrollado este TFG
SVN	Repositorio de Versiones
Django Mobile Client	La "parte del terminal" del sistema *Django*
Workflow	Flujo de trabajo de la aplicación
Finished-order	Mensaje xml utilizado para sincronizar información entre el terminal y el servidor.
Backend	El "lado del servidor" de un sistema
Django Central	Base de datos central de *Django*

Anexos

A Ejemplo Albarán Venta en Ruta

DOCUMENTO DE CIRCULACION
ALBARAN DE VENTA EN RUTA - CARTA DE PORTE

ALBARAN CIRCUL. NUM: 16RCA000070
FECHA: 20/08/2016
EXPEDIDOR: COMPAÑIA ESPAÑOLA DE PETROLEOS
NIF: A28003119
ALMACEN: DEP.CAZALEGAS (TO)-CC CENTRO
CAE: ES0045HTC05R
DIRECCION: Autovia de Extremadura, N-V 45683 Cazalegas

SUMINISTRADOR: COMPAÑIA ESPAÑOLA DE PETROLEOS
NIF: A28003119
CARGADOR CONTRACTUAL: LA PETROLÍFERA TRANSPORTES, S
NIF: A28017E45
TRANSPORTISTA EFECTIVO: LA PETROLIFERA DE CASTILLA
NIF: B86835170
CONDUCTOR: ANGEL RODRIGUEZ MANZANEQUE
VEHICULO: ESCRB375N

CARGA EN ALMACEN

TIPO	VL(L)	T(ºC)	V15(L)
G0CR	600,00L	12,00	601,43L
G0A	1.500,00L	11,00	1.504,53L
G0B	3.600,96L	11,00	3.611,83L

CARGA POR COMPARTIMENTO

DP	PROD	VL(L)	T(ºC)	V15(L)
DP1	G0B	3.600,96L	11,00	3.611,83L
DP2	G0A	1.500,00L	11,00	1.504,53L
DP3	G0CR	600,00L	12,00	601,43L

NUM: D1680338740
NIF: ESA45437111
CLIENTE: GESTION MEDIOAMBIENTAL DE TOLEDO SA
DIRECCION: DE GRECIA 1 PTAL 2 OFI 20
LOCALIDAD: 45005 TOLEDO
Gasoleo A 1.400,00L
EXISTENCIAS: 100,00L

NUM: D1680332790
NIF: ES70408872K
CLIENTE: RODRIGUEZ MUÑOZ GALDENCIO
DIRECCION: PUENTE SN
LOCALIDAD: 45661 ALDEANUEVA DE BARBARROYA
Gasoleo B 200,00L
EXISTENCIAS: 3.400,96L

TOTAL SUMINISTRADO

TIPO	VL(L)
Gasoleo C Rendim	0,00L
Gasoleo A	1.400,00L
Gasoleo B	200,00L

DEVOLUCION A ALMACEN

TIPO	VL(L)	T(ºC)	V15(L)
Gasoleo C Rendim	600,00L	0	0,00L
Gasoleo A	100,00L	0	0,00L
Gasoleo B	3.400,96L	0	0,00L

FECHA: 20/08/2016
CARTA DE PORTE EN VACIO:
A efectos de cumplimiento del A.D.R., se declara:
Vehiculo cisterna vacio.
Ultima mercancia cargada la que figura en la primera parte de este documento.
El conductor:

CABECERA DE VENTA EN RUTA

SUMINISTROS REALIZADOS

PIE DE VENTA EN RUTA

B Ejemplo Nota de Entrega

NOTA DE ENTREGA

NOTA ENTREGA: 01660338740
ALBARAN CIRCUL. NUM:
FECHA: 21/01/2016
EXPEDIDOR: COMPAÑIA ESPAÑOLA DE PETROLEOS
NIF: A28003119
ALMACEN: DEP.CAZALEGAS (TO)-CC CENTRO
CAE: ES00045HT069R
DIRECCION: Autovia de Extremadura, N-V 45683 Cazalegas

SUMINISTRADOR: COMPAÑIA ESPAÑOLA DE PETROLEOS
NIF: A28003119
CONDUCTOR: ANGEL RODRIGUEZ MANZANEQUE
VEHICULO: ESCR8375W
CLIENTE: GESTION MEDIOAMBIENTAL DE TOLEDO SA
NIF: ESA45407111
CODIGO: 0020101509
DIRECCION: DE GRECIA 1 PTAL 2 OFI 20
LOCALIDAD: 45005 TOLEDO

Gasoleo A	1.400,00L
PRECIO FINAL:	0,65 EUR/L
TOTAL A PAGAR:	1.105,34 EUR

***EXCLUSIVAMENTE A EFECTOS FISCALES**

T(ºC)*	**V15(L)***
15,00*	1.400,00L*

OBSERVACIONES:
COBRADO: 1.105,34

FIRMA CONDUCTOR **FIRMA CLIENTE**

C Caudalímetro